

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

Daniela de Oliveira Preto

**RECONHECIMENTO DE PLACAS DE TRÂNSITO POR  
MEIO DE DEEP LEARNING**

Florianópolis

2018



Daniela de Oliveira Preto

## **RECONHECIMENTO DE PLACAS DE TRÂNSITO POR MEIO DE DEEP LEARNING**

Trabalho de conclusão de curso submetido ao Programa de graduação em Ciências da Computação para a obtenção do Grau de Bacharela em Ciências da Computação.

Orientador: Prof. Dr. Mauro Roisenberg

Florianópolis

2018

Catálogo na fonte elaborada pela biblioteca da  
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:

<http://www.bu.ufsc.br/design/Catalogacao.html>

Daniela de Oliveira Preto

## **RECONHECIMENTO DE PLACAS DE TRÂNSITO POR MEIO DE DEEP LEARNING**

Este Trabalho de conclusão de curso foi julgado aprovado para a obtenção do Título de “Bacharela em Ciências da Computação”, e aprovado em sua forma final pelo Programa de graduação em Ciências da Computação.

Florianópolis, 30 de novembro 2018.

---

Prof. Dr. Eng. Rafael Luiz Cancian  
Coordenador do Curso

### **Banca Examinadora:**

---

Prof. Dr. Mauro Roisenberg  
Orientador

---

Prof. Dr. Elder Rizzon Santos

---

Profa. Dra. Jerusa Marchi



"Dedico este trabalho à minhas avós Maria (In Memoriam) e Imaculada (In Memoriam), meus maiores exemplos de vida, com muito amor e saudade."





## AGRADECIMENTOS

Agradeço inicialmente aos meus pais, Ana Maria e Carlos Augusto, e minha irmã Simone, pelo amor, paciência e apoio em todos os momentos, mas em especial nesta jornada acadêmica.

Aos meus irmãos de coração, Amanda Teixeira, Bruno Manica e Carolina Petruso, por todos os momentos e conversas que me fizeram não desistir e ainda me deram forças para continuar.

A todos os amigos e colegas que participaram e me ajudaram de alguma forma na minha vida e graduação, em especial aos amigos da Atlética do CTC, da Atlética da Computação, do centro acadêmico da computação e dos times de futsal do CTC e da UFSC, vocês me fizeram vivenciar os melhores anos de minha vida.

Ao meu orientador Mauro Roisenberg, por me aceitar como sua orientanda e me ajudar sempre da melhor maneira possível.

Aos professores do Departamento de Informática e Estatística, por toda a dedicação em consagrar o curso de ciências da computação da UFSC como um dos melhores do Brasil.

A Universidade Federal de Santa Catarina, por me acolher e me proporcionar uma experiência de vida única.

A todos vocês, meus sinceros agradecimentos.



*"Não espere o futuro mudar tua vida, porque o futuro é a consequência do presente."*

Racionais Mc's



## RESUMO

A inteligência artificial está cada vez mais presente em nosso cotidiano, sendo em equipamentos médicos, em sistemas de música por streaming, em buscadores na web e até em automóveis. Este projeto terá como foco um subgrupo da inteligência artificial conhecido como *Deep Learning* (Aprendizagem Profunda). Este subgrupo é um método de trabalho desenvolvido a partir do avanço da neurociência, onde o sistema de aprendizado da máquina é baseado no que se conhece do sistema nervoso humano, tentando computar dados de entradas como se fossem estímulos de um cérebro, e analisando as respostas que serão geradas em cada caso.

A técnica de *Deep Learning* pode ser descrita como um modelo em camadas onde, em cada camada, o algoritmo vai extraindo informações sobre a entrada que foi fornecida. Para a primeira camada é usada a entrada em si, e as demais utilizarão a saída da anterior. Este modelo também é conhecido como Redes Neurais Artificiais Profundas. Neste projeto este método será usado para o reconhecimento de placas de trânsito, fazendo com que sejam analisados os pixels da imagem para assim classificá-la. Nas diversas camadas serão analisadas características como o formato e as cores. O objetivo final é poder analisar e classificar corretamente as imagens, mesmo que tenham diferença de luminosidade ou um fundo diferente.

**Palavras-chave:** Inteligência Artificial. Aprendizado de Máquina. Aprendizagem Profunda. Redes Neurais.



## ABSTRACT

Artificial intelligence is increasingly in our daily lives, being in medical equipment, streaming music systems, in web search engines and even in cars. This project will focus on the subgroup of artificial intelligence known as Deep Learning. This subgroup is a working method developed from the advance of neuroscience, where the learning system of the machine is based on what is known of the nervous system, trying to compute input data as if it were stimulus of a brain, and analyzing the responses that will be generated in each case.

Deep Learning can be described as a layer model, where in each layer the algorithm extracts information on the entry that was provided. For the first layer the input itself is used and the others will use the output of the previous one. In this project will be used this method for the recognition of traffic signs, that the pixels of the image are analyzed and then classify it. In the many layers will be considered features such as shape and colors. The final objective is to be able to analyze and classify correctly the images, even if they have a difference in brightness or a different background.

**Keywords:** Artificial intelligence. Machine Learning. Deep Learning. Neural Networks





## LISTA DE FIGURAS

Figura 1	Representação de uma rede neural artificial .....	28
Figura 2	Representação de um neurônio artificial.....	28
Figura 3	Desempenho x Quantidade de dados.....	30
Figura 4	Comparação da taxa de erro ao longo dos anos .....	31
Figura 5	Representação de entradas de uma convolução.....	32
Figura 6	Representação dos cálculos de uma convolução.....	33
Figura 7	Representação do cálculo de saída de uma convolução..	34
Figura 8	Kernel referente a imagem RGB.....	35
Figura 9	Representação: (a) da entrada; (b) do kernel.....	35
Figura 10	Representação de como obter a saída da camada convo- lucional. ....	36
Figura 11	Representação de um Pooling.....	37
Figura 12	Representação da camada de entrada.....	38
Figura 13	Representação da camada totalmente conectada .....	39
Figura 14	Representação do algoritmo de RCNN .....	40
Figura 15	Representação da saída do algoritmo de HOG .....	41
Figura 16	Exemplo de detecção: (a,b) imagens de entrada; (c,d) detecção das possíveis regiões pelo formato; (e,f) possíveis áreas utilizando os filtros de cores. ....	43
Figura 17	Exemplo de Segmentação por Limiarização em cenários de trânsito da Alemanha. ....	44
Figura 18	Exemplos de segmentação por fuzzificação em cenários de trânsito do Brasil. Para cada cenário é apresentada sua imagem segmentada com valores de intensidade pintados da cor de interesse, para fins de ilustração. ....	45
Figura 19	Detecção utilizando o modelo HSB .....	46
Figura 20	Exemplo de imagens de entrada.....	50
Figura 21	Treinamento rede A .....	52
Figura 22	Treinamento rede B .....	52
Figura 23	Treinamento rede C .....	53
Figura 24	Treinamento rede D .....	53
Figura 25	Treinamento rede E .....	53
Figura 26	Treinamento rede F .....	53

Figura 27	Exemplos de imagens com regiões de interesse .....	54
Figura 28	Exemplos de "negativos" .....	55
Figura 29	Regiões de interesse após o algoritmo de HOG .....	55
Figura 30	Matriz de confusão .....	63
Figura 31	Exemplos de saída do algoritmo 3.....	64
Figura 32	Teste de RCNN com foto .....	65
Figura 33	Teste de RCNN com imagem em tempo real .....	66
Figura 34	Teste de detecção usando HOG .....	66

## LISTA DE TABELAS

Tabela 1	Camadas utilizadas para teste .....	47
Tabela 2	Redes neurais implementadas.....	52



## LISTA DE ABREVIATURAS E SIGLAS

PNG	Portable Network Graphics .....	25
RNA	Redes Neurais Artificiais .....	27
CNN	Convolutional Neural Network .....	32
RGB	Red-Green-Blue.....	34
ReLU	Rectified Linear Unit.....	36
RCNN	Region-based Convolutional Neural Network .....	40
HOG	Histograms of Oriented Gradients.....	40
HSB	Hue, Saturation e Brightness.....	46



## SUMÁRIO

<b>1 INTRODUÇÃO</b>	23
1.1 MOTIVAÇÃO	23
1.2 OBJETIVO	24
1.2.1 Objetivo Geral	24
1.2.2 Objetivos Específicos	24
1.3 METODOLOGIA	24
1.3.1 Revisão da literatura	25
1.3.2 Banco de Imagens	25
1.3.3 Ferramenta	25
1.4 ESTRURA DO TRABALHO	26
<b>2 FUNDAMENTAÇÃO TEÓRICA</b>	27
2.1 REDES NEURAIS ARTIFICIAIS (RNA)	27
2.1.1 Neurônios Artificiais	28
2.2 <i>DEEP LEARNING</i>	29
2.2.1 <i>Backpropagation</i>	31
2.3 REDES NEURAIS CONVOLUCIONAIS	32
2.3.1 Convolução	32
2.3.2 Camada Convolutacional	34
2.3.3 Camada de retificação	36
2.3.4 Camada de <i>Pooling</i>	37
2.3.5 Camada de <i>SoftMax</i>	37
2.3.6 Camada de classificação	38
2.3.7 Camada de entrada	38
2.3.8 Camada totalmente conectada	39
2.4 DETECÇÃO DE REGIÕES DE INTERESSE	39
2.4.1 Redes Neurais Convolucionais baseadas em regiões	40
2.4.2 Histogramas de gradientes orientados	40
<b>3 TRABALHOS CORRELATOS</b>	43
3.1 DETECÇÃO AUTOMÁTICA E RECONHECIMENTO DE SINALIZAÇÃO DE TRANSITO USANDO ANALISE DE ESTRUTURAS GEOMÉTRICAS	43
3.2 DETECÇÃO E CLASSIFICAÇÃO DE SINALIZAÇÃO VERTICAL DE TRÂNSITO EM CENÁRIOS COMPLEXOS	44
3.3 EXTRAÇÃO DE INFORMAÇÕES DE COR E FORMA PARA DETECÇÃO DE PLACAS DE TRÂNSITO EM IMAGENS	46

3.4 REDES CONVOLUCIONAIS MUITO PROFUNDAS PARA RECONHECIMENTO DE IMAGENS EM LARGA ESCALA	47
<b>4 METODOLOGIA PROPOSTA</b>	49
4.1 DADOS DE ENTRADA	49
4.2 CONFIGURAÇÕES E PARÂMETROS	50
4.3 ARQUITETURAS DE CNN	51
4.4 DETECÇÃO	54
4.4.1 Treinamento da RCNN	54
4.4.2 Treinamento do HOG	54
<b>5 IMPLEMENTAÇÃO</b>	57
5.1 CRIAÇÃO DE UMA CNN	57
5.2 DETECÇÃO DE IMAGENS E CLASSIFICAÇÃO	59
<b>6 RESULTADOS</b>	63
6.1 RESULTADOS DE CLASSIFICAÇÃO	63
6.2 RESULTADOS DE CLASSIFICAÇÃO COM DETECÇÃO	65
<b>7 CONCLUSÃO</b>	67
7.1 TRABALHOS FUTUROS	67
<b>REFERÊNCIAS</b>	69
<b>APÊNDICE A – Fontes</b>	75
<b>APÊNDICE B – Artigo</b>	77



# 1 INTRODUÇÃO

Com o aumento significativo de automóveis ao redor no mundo, a sinalização de trânsito ganhou um papel importante para evitar incidentes automobilísticos (GLOBAL STATUS REPORT ON ROAD SAFETY, 2015). As placas de trânsito foram criadas para que tenham fácil visibilidade, legibilidade e precisão, de maneira que o motorista consiga facilmente saber o que deve ser feito (DENATRAN, 2000).

Segundo o observatório Nacional de Segurança Viária, 90 % dos acidentes são causadas por falhas humanas, fazendo com que cada vez mais se invistam em sistemas inteligentes para automóveis (ZANNI, 2014).

Desde a segunda metade do século passado ouvimos falar de inteligência artificial. (RUSSELL; NORVIG, 2013) Um tema que era tratado como futurista em filmes e documentários se tornou realidade, sendo hoje um assunto bastante comentado no mundo todo. Três fatores foram decisivos para estarmos no patamar atual, o avanço do poder de processamento dos computadores, a grande quantidade de dados disponíveis e o desenvolvimento de técnicas e algoritmos. (CHEN; LIN, 2014)

Este trabalho fala especificadamente do terceiro fator e como a partir destas técnicas e algoritmos podemos realizar o reconhecimento de imagens, especificamente de placas de trânsito, através de Deep Learning, podendo assim auxiliar pessoas quando estiverem dirigindo.

## 1.1 MOTIVAÇÃO

Ferramentas e técnicas avançadas melhoraram drasticamente os algoritmos de aprendizado de máquina, ao ponto de que ele possa superar os seres humanos na classificação de imagens, ganhar do melhor jogador de "GO" e apenas por controle de voz fazer com que o Google Home encontre e baixe a nova música que você gosta. (MATLAB; SIMULINK, 2017)

Motivada por estes avanços e pelo aumento do uso de reconhecimento de imagens tanto em áreas como a biomedicina, que é capaz de descobrir a morfologia de uma célula (ANDREY, 2017), quanto em áreas como o transporte, com a invenção dos carros autônomos fazendo com que o uso de reconhecimento de imagens seja essencial para reconhecer o ambiente a sua volta, conseguindo distinguir objetos, pessoas e

outros automóveis, dentro deste grupo de objetos temos as placas de transito.(JANAI et al., 2017)

## 1.2 OBJETIVO

### 1.2.1 Objetivo Geral

Este projeto tem o objetivo construir um modelo que possa reconhecer placas de trânsito por meio do reconhecimento dos diversos padrões encontrados. Sendo capaz ao final desde projeto de classificar uma dada placa de trânsito capturada através de uma câmera.

### 1.2.2 Objetivos Específicos

Para se chegar ao objetivo geral, serão necessários objetivos específicos, listados a seguir:

- Realizar estudo sobre os conceitos de Redes Neurais Artificiais e de *Deep Learning*.
- Compreender o relacionamento das Redes Neurais Artificiais com Deep Learning e realizar estudo sobre as Redes Neurais Artificiais mais utilizadas para reconhecimento de imagens.
- Criar uma aplicação em MatLab para o reconhecimento de placas de transito.
- Realizar testes utilizando diferentes redes neurais tendo placas de trânsito como dado de entrada.
- Classificar uma placa de trânsito podendo essa ser capturada por uma câmera, podendo ser foto ou vídeo.
- Avaliar os resultados obtidos.

## 1.3 METODOLOGIA

Este trabalho foi desenvolvido em três fases: revisão da literatura, criação de um banco de imagens de placas de sinalização brasileiras categorizadas por seus respectivos significados e implementação

de uma rede neural profunda, em MATLAB, capaz de reconhecer tais placas.

### 1.3.1 Revisão da literatura

A revisão da literatura foi baseada em trabalhos e artigos sobre redes neurais artificiais, aprendizado profundo e visão computacional, publicados por estudantes e pesquisadores das mais diversas universidades ao redor do mundo. A pesquisa bibliográfica também incluiu trabalhos e a documentação realizada pelos desenvolvedores do software MATLAB.

### 1.3.2 Banco de Imagens

Para que esse trabalho fosse realizado, foi preciso um banco de imagens de placas de sinalização para serem usadas como dados de treinamento e validação da rede neural.

Uma vez que só foram encontrados bancos prontos com placas de sinalização estrangeiras, os quais mesmo seguindo as convenções de trânsito viário de Viena acabam sendo diferentes de país para país (DETRAN, 1978), em razão disso foi montado um banco de imagens capturadas por meio do *Google Street View*, recurso desenvolvido e disponibilizado pela empresa Google LLC.

Contudo, segundo o Termo De Serviço Adicional Do Google Maps, 2018, é considerada conduta proibida a redistribuição ou revenda, portanto o banco não ficará disponível para *download*.

Foram capturadas 4078 imagens no formato PNG, de 128 *pixels* de altura por 128 de largura, onde 80% será usado para o treinamento da rede e 20% para teste. Essas imagens também foram separadas em 12 categorias, cada uma de acordo com o seu respectivo significado.

### 1.3.3 Ferramenta

Esse trabalho foi desenvolvido com um processador Intel(R) *core(TM)* *i7-5500u CPU @ 2.40ghz* e uma placa de vídeo dedicada de 2GB Intel(R) *HD Graphics 5500*.

A ferramenta utilizada para o desenvolvimento deste trabalho foi o MatLab, criado pela MathWorks Inc, um ótimo software para análise de dados e tendo como um grande ponto positivo o suporte para *Deep*

*Learning* e visão computacional.

## 1.4 ESTRURA DO TRABALHO

Este trabalho está dividido em seis capítulos além da introdução, sendo eles:

- Fundamentação teórica: Explica detalhadamente os conceitos prévios importantes para o entendimento do trabalho desenvolvido;
- Trabalhos correlatos: Apresenta pesquisas desenvolvidas relacionadas ao tema desde trabalho;
- Modelo Proposto: Baseado na fundamentação teórica, mostra-se como foram montadas as arquiteturas de redes neurais para classificação e a extração de características para a detecção;
- Implementação: Mostra trechos importantes para a elaboração deste trabalho
- Resultados: Realiza-se uma análise dos resultados obtidos;
- Conclusão: Mostra-se qual foi a conclusão da pesquisa e a partir desse ponto, quais os possíveis trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Para um melhor entendimento deste projeto, neste capítulo serão explicados conceitos importantes para o entendimento de *Deep Learning*, redes neurais artificiais e suas variações.

### 2.1 REDES NEURAIIS ARTIFICIAIS (RNA)

Com o interesse científico de entender o funcionamento do cérebro, de como, a partir de experiências e de repetidos estímulos, os humanos conseguem aprender, houve um avanço no campo da neurociência e no entendimento das redes neurais. (FAUSETT, 1994)

Segundo a Data Science Academy, 2017, o desenvolvimento do cérebro humano ocorre principalmente nos dois primeiros anos de vida, mas se arrasta por toda a vida. Inspirando-se neste modelo, diversos pesquisadores tentaram simular o funcionamento do cérebro, principalmente o processo de aprendizagem por experiência, a fim de criar sistemas inteligentes capazes de realizar tarefas como classificação, reconhecimento de padrões, processamento de imagens, entre outras atividades. Como resultado destas pesquisas surgiu o modelo do neurônio artificial e posteriormente um sistema com vários neurônios interconectados, a chamada Rede Neural.

Uma rede neural é uma máquina projetada para modelar a maneira pela qual o cérebro executa uma tarefa ou função particular de interesse (HAYKIN, 2009)

Ela pode ter uma ou várias camadas. As redes que possuem uma única camada são as redes que possuem um nó entre uma entrada e uma saída da rede. Já as redes multicamadas possuem uma ou mais camadas entre entrada e saída, e essas camadas são chamadas de camadas ocultas. As redes multicamadas são utilizadas para a aprendizagem profunda e quanto mais camadas houver, mais profunda será a rede.(ZAMBIASI, 2008)

Na figura 1, podemos ver a representação de uma rede neural. No caso de reconhecimento de imagens, cada neurônio da camada de entrada representaria um pixel. Nas camadas ocultas, ocorreriam os cálculos para o reconhecimento. Na camada de saída, cada neurônio representaria uma possibilidade de reconhecimento. Como nesta imagem temos apenas um neurônio na saída, apenas saberíamos se a imagem representa ou não um dado objeto.(NIELSEN, 2015)

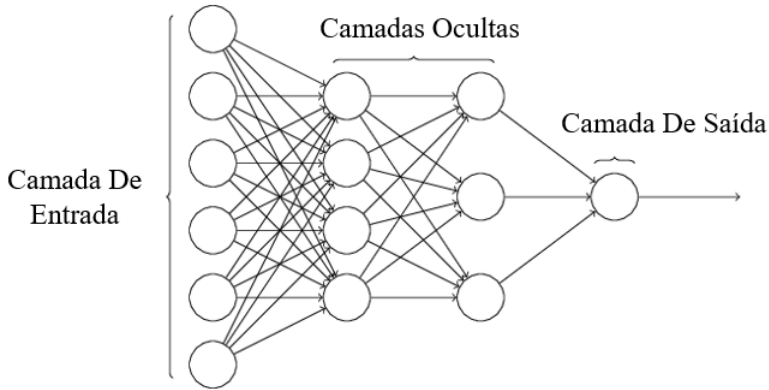


Figura 1 – Representação de uma rede neural artificial

Fonte: Nielsen, 2015

### 2.1.1 Neurônios Artificiais

O neurônio artificial é uma unidade de processamento utilizada para fazer operações de uma RNA. O modelo mais aceito foi proposto nos anos 40 por McCulloch e Pitts, no qual o neurônio calcula a soma ponderada de várias entradas, aplica uma função de ativação e passa o resultado para a próxima camada. Esse modelo está ilustrado na figura 2, e segundo Haykin, 2009, é composto pelos seguintes elementos:

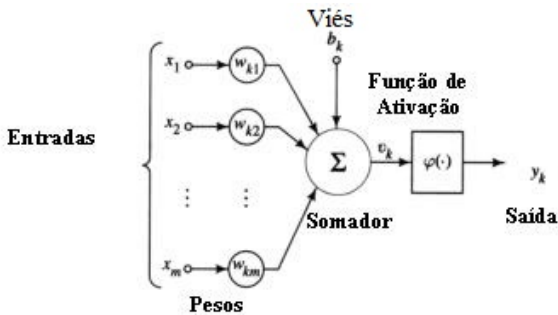


Figura 2 – Representação de um neurônio artificial

Fonte: HAYKIN (Traduzido), 2009

- Sinais de entrada ( $x_1, x_2, \dots, x_n$ ):

São os sinais externos normalmente normalizados para incrementar a eficiência computacional dos algoritmos de aprendizagem. Quando normalizados recebem valores entre 0 e 1.

- Pesos sinápticos( $w_1, w_2, \dots, w_n$ ):

São valores para ponderar os sinais de cada entrada da rede, dentre os vários estímulos recebidos, alguns excitarão mais e outros menos o neurônio receptor. Esses valores são aprendidos durante o treinamento.

- Combinador linear ( $\sum$ ):

Multiplica todos sinais de entrada  $x_n$  pelos respectivos pesos sinápticos  $w_n$  a fim de produzir um potencial de ativação. Pode ser representado com a multiplicação das matrizes  $|X| * |W|$ .

- Limiar de ativação ou viés ( $b_k$ ):

Especifica qual será o nível apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de ativação.

- Potencial de ativação ( $v_k$ ):

É o resultado obtido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Se o valor for positivo, ou seja, se  $u \geq 0$  então o neurônio produz um potencial excitatório; caso contrário, o potencial será inibitório.

- Função de ativação ( $\gamma$ ):

É um elemento extremamente importante das RNA. Ela basicamente decide se um neurônio deve ser ativado ou não, de acordo com o valor  $v_k$ . Ou seja, se a informação que o neurônio está recebendo é relevante ou deve ser ignorada.

- Sinal de saída ( $y_k$ ):

É o valor final de saída que será usado como entrada de outros neurônios que estão sequencialmente interligados.

## 2.2 DEEP LEARNING

Nos últimos anos, o interesse pelo aprendizado de máquina ou mais conhecido como *Machine Learning*, aumentou muito. O *Machine*

*Learning* é o uso de algoritmos para retirar informações de dados brutos e transformá-los em tendências e previsões refinadas (Udacity, 2017).

Existem vários algoritmos que exercem essa função, mas o que mais tem se destacado são as redes neurais artificiais profundas, ou também conhecidas como *Deep Learning*, pois não necessitam de extração manual de características. Como já dito na sessão 2.1, esse modelo emprega algoritmos para processar dados imitando o processamento do cérebro humano. Enquanto uma RNA tradicional possui apenas 2 ou 3 camadas ocultas, uma RNA profunda pode possuir centenas (MATLAB Documentation, 2017).

Uma RNA profunda é muito mais difícil de treinar que uma simples RNA, porém para um conjunto maior de dados de entrada, possui uma maior eficiência. (NIELSEN, 2015)

Como já dito, em outros modelos de aprendizado de máquina, as características, mais comumente chamadas de *features*, precisam ser extraídas manualmente, porém no modelo de *Deep Learning* o aprendizado é de início ao fim (*end-to-end learning*) a partir do dado de entrada bruto. Sua principal vantagem é que o aprendizado de uma rede profunda vai melhorando a medida que os dados de entrada crescem, ao passo que em outros algoritmos, fica estabilizado (MATLAB Documentation, 2017). Na figura 3 é mostrado graficamente essa vantagem.

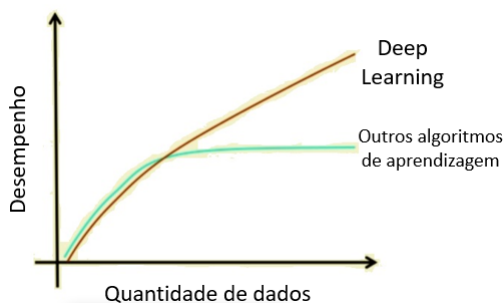


Figura 3 – Desempenho x Quantidade de dados

Fonte: Towards Data Science (Traduzido)

A aprendizagem profunda é responsável por grandes avanços na área de visão computacional, enquanto outros modelos reconhecem com facilidade um dado objeto, um segundo objeto, parcialmente obstruído, na mesma imagem, teria grande chance de não ser reconhecido ou ser



reconhecido de forma equivocada.(PEREIRA, 2017)

Na figura 4, podemos ver que ao longo dos anos, com o aperfeiçoamento da aprendizagem profunda, a taxa de erros para o reconhecimento de imagens de uma rede treinada é similar a taxa de erros de um ser humano.

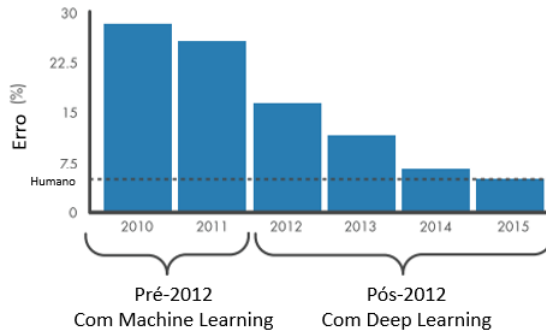


Figura 4 – Comparação da taxa de erro ao longo dos anos  
 Fonte: The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

### 2.2.1 *Backpropagation*

Segundo a Data Science Academy, 2017, o algoritmo de backpropagation é considerado o principal algoritmo da história das redes neurais, sem ele não existiriam as redes profundas como as conhecemos. Seu principal objetivo é otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas para as saídas. Ele consiste em duas fases, a primeira é conhecida como *Forward Pass*. É responsável por propagar a entrada de acordo com os neurônios que vão sendo ativados na passagem, até que chegue na camada de saída.

A segunda fase é conhecida como *Backward Pass*, onde é calculado o gradiente da função de perda na camada final e esse gradiente é usado para atualizar recursivamente os pesos da rede treinada. Esse passo também é conhecido como retro-propagação.

## 2.3 REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais, também são conhecidas como ConvNet ou CNN (Convolutional Neural Network). (Data Science Academy, 2017). Como uma RNA tradicional as redes neurais convolucionais também possuem neurônios com pesos, vieses e funções não lineares, porém possui grande vantagem quando utilizada em imagens, por possuir o poder de codificar propriedades, faz com que o treinamento diminua e a precisão seja mantida. (PACHECO, 2017)

### 2.3.1 Convolução

A convolução é uma operação matemática que consiste em a partir de duas funções ou sinais, no qual a segunda tem seus sinais trocados, gerar um produto.

Na figura 5 temos duas representações de sinais, X (em laranja) e W (em verde). O primeiro passo na convolução é a partir de W, gerar um W invertido que chamaremos de W', o qual está representado em azul.

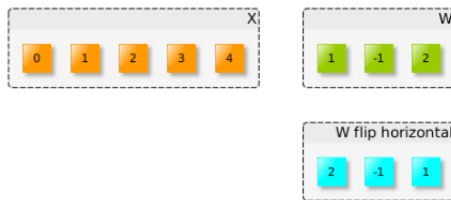


Figura 5 – Representação de entradas de uma convolução

Fonte: Leonardo Araujo Santos

Na figura 6 podemos ver os próximos passos da convolução. No item 1, é multiplicado o primeiro valor de X pelo último valor de W', e assim W' é movido uma posição gerando o que temos no item 2, assim o primeiro valor de X multiplica o penúltimo valor de W' e o resultado é somado com segundo valor de X multiplicado pelo último valor de W'. No item 3, W' é movido mais uma posição e outras multiplicações são realizadas. Isso ocorre até que o último número de X seja multiplicado pelo primeiro valor de W', mostrado no item 7.

Para utilizar esse conceito em uma CNN, os valores de X repre-

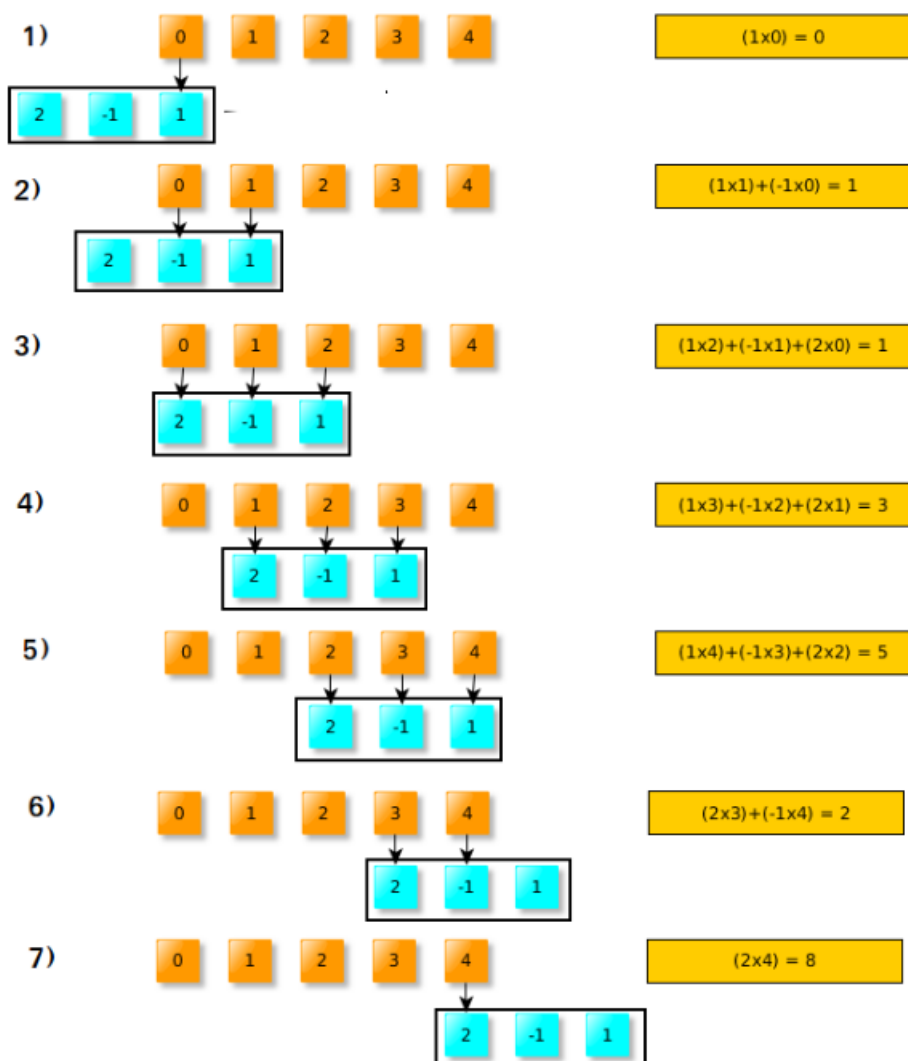


Figura 6 – Representação dos cálculos de uma convolução  
 Fonte: Leonardo Araujo Santos

sentam pixels da imagem enquanto os valores de  $W'$  são os chamados pesos (weights), para o uso em imagens são conhecidos como filtros (*filters*) ou *kernel*. Para gerar um resultado válido, apenas são consideradas as operações em que todos os pesos estão multiplicando um dos pixels de entrada, mostrado na imagem 7 como *valid cases*. Também na figura 7 podemos ver que o resultado é dado pelo número de produtos válidos, mostrados em *Result*.

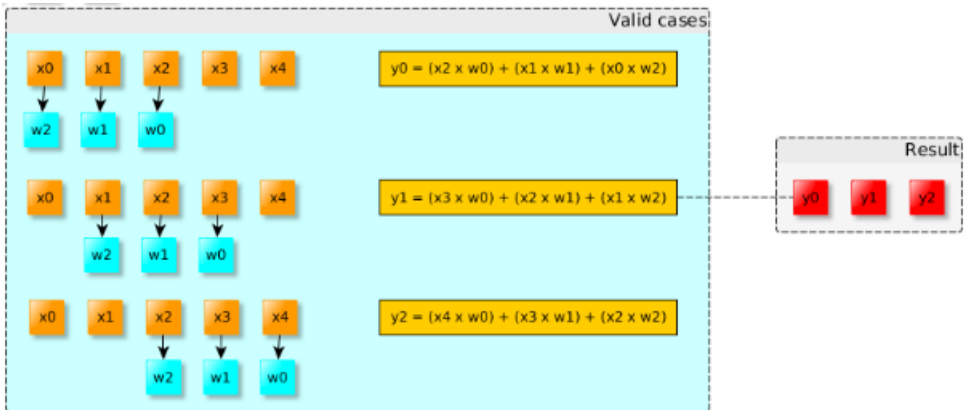


Figura 7 – Representação do cálculo de saída de uma convolução  
Fonte: Leonardo Araujo Santos

### 2.3.2 Camada Convolutacional

Essa camada utiliza o conceito de convolução, porém pelo fato de uma imagem possuir duas dimensões, o filtro será uma matriz bidimensional. Em caso de imagens utilizando a escala de cinza, a matriz terá apenas um canal, e será bidimensional. Em casos onde a entrada é uma imagem na escala RGB, a entrada terá 3 canais, tornando a matriz tridimensional, como mostrado na figura 8.

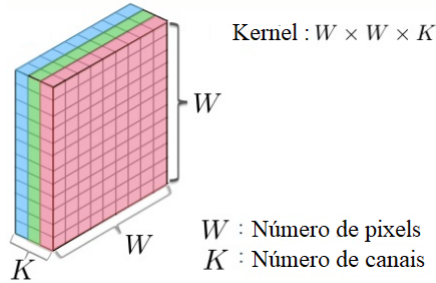


Figura 8 – Kernel referente a imagem RGB  
 Fonte: Ken'ichi Matsui, 2015

Nas próximas duas figuras, temos um exemplo de como a camada funciona. Na figura 9 temos uma matriz 3x3 representando a entrada e uma matriz 2x2 representando o kernel. Na figura 10 é descrito passo a passo das operações para que seja gerado o resultado, que será no caso a saída da camada convolucional.

1	3	1
0	-1	1
2	2	-1

(a)

-1	0
2	1

(b)

Figura 9 – Representação: (a) da entrada; (b) do kernel  
 Fonte: Leonardo Araujo Santos

A seguir tem-se a definição de componentes importantes para o uso da camada convolucional. O primeiro é o conceito de *Stride*, do inglês "passo", que nada mais é do que o movimento que o filtro irá fazer sobre a imagem. Caso o Stride seja 1, o movimento será apenas de um pixel para o lado, caso seja 2, o filtro se movimentará 2 pixels para o lado.

O conceito de *Zero Padding* ou preenchimento zero na tradução literal. Foi feito para controlar o tamanho da saída da camada, adicionando zeros em toda a borda da matriz de entrada. Para se obter o tamanho da saída é utilizada a seguinte fórmula

$$\frac{(t - ((f - 1) * d + 1) + 2 * p)}{s} + 1 \quad (2.1)$$

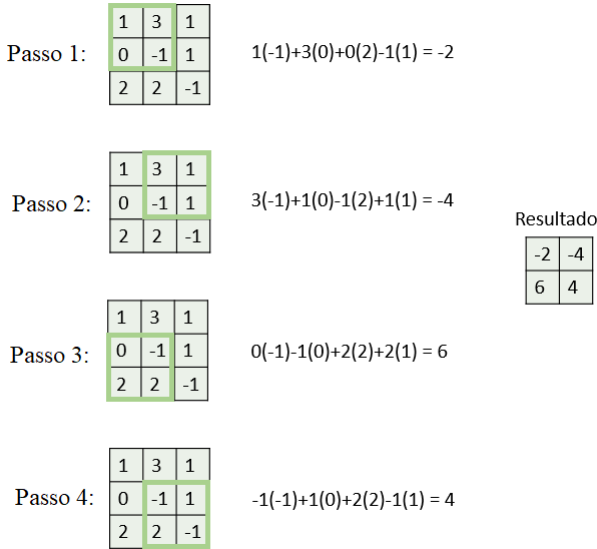


Figura 10 – Representação de como obter a saída da camada convolucional.

Fonte:Leonardo Araujo Santos

sendo "t"o tamanho da entrada, "f"o tamanho do filtro, "d"o fator de dilatação, "p"o *padding* e "s"o *stride*.

Outro conceito é o fator de dilatação, um vetor de dois números inteiros, o primeiro para altura e o segundo para largura, usado para aumentar a área da entrada que a camada consegue ver, porém sem aumentar os parâmetros computacionais.

### 2.3.3 Camada de retificação

A camada de retificação, também conhecida como ReLu (*Rectified Linear Unit*), ela normalmente é utilizada após a camada convolucional criando um limite para cada elemento da entrada de acordo com a seguinte função

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.2)$$

### 2.3.4 Camada de *Pooling*

A camada de *Pooling* é usada para reduzir a altura e largura da entrada, porém a profundidade não se altera. Com o uso dessa camada a performance aumenta pois há menos informações a serem computadas. Existe dois tipos de *pooling*, em ambos a entrada é dividida em regiões, porém no *Max Pooling* o resultado é o maior valor da região, enquanto no *Average* (tradução do inglês: média) *Pooling* é calculado o valor médio de cada região.

Na figura 11 está exemplificado utilizando *Max Pooling*, onde na região mostrada em rosa o maior valor é o 6, fazendo com que esse seja o valor na saída. Na região verde será o 8, na amarela o 3 e na azul o 4, fazendo com que a saída seja a matriz mostrada do lado direito.

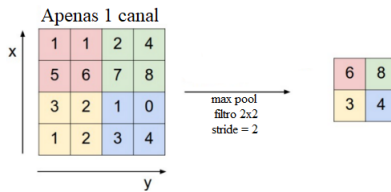


Figura 11 – Representação de um Pooling  
Fonte: Leonardo Araujo Santos (Traduzido)

### 2.3.5 Camada de *SoftMax*

O objetivo dessa camada é obter a probabilidade de a imagem de entrada pertencer a uma dada classe, por esse motivo ela é utilizada após a (s) camada (s) totalmente conectada (s). A função de SoftMax é dada por

$$P(c_r|x, 0) = \frac{P(x, 0|c_r)P(c_r)}{\sum_{j=1}^k P(x, 0|c_j)P(c_j)} \quad (2.3)$$

onde  $0 \leq P(c_r|x, 0) \leq 1$ ,  $\sum_{j=1}^k P(x, 0|c_j)P(c_j) = 1$ ,  $P(x, 0|c_r)$  é a probabilidade condicional de uma dada classe  $r$ , e  $P(c_r)$  é a probabilidade prévia de pertencer a uma classe  $r$ .

### 2.3.6 Camada de classificação

Essa camada utilizada em problemas de classificação, calcula a perda de entropia das classes de saída, pois essas são mutualmente exclusivas. Tem como entrada os valores recebidos da camada *SoftMax* e atribui às K classes de saída, a perda é calculada utilizando a seguinte função

$$loss = \sum_{i=1}^N \sum_{j=1}^K t_{ij} \ln y_{ij} \quad (2.4)$$

onde N é o número de amostras, K é o número de classes,  $t_{ij}$  é a indicação de que a i-ésima amostra pertence a j-ésima classe, e  $y_{ij}$  é o valor recebido da camada anterior.

### 2.3.7 Camada de entrada

Percorre o dado de entrada colocando cada valor ou pixel em um neurônio da camada e também verifica o número de canais que está sendo usado.

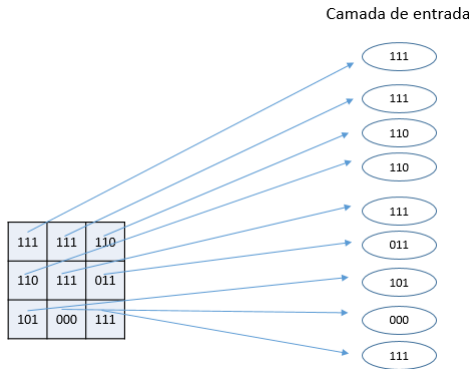


Figura 12 – Representação da camada de entrada

Fonte: Daniela Preto



### 2.3.8 Camada totalmente conectada

Como o nome já diz, cada neurônio dessa camada está conectado a todas as saídas da camada anterior. Dado que a camada que a precede tenha  $n$  saídas, a camada totalmente conectada terá  $n$  neurônios.

Nesse momento todas as informações que foram previamente aprendidas por outras camadas são utilizadas para identificar padrões, quando a informação  $X$  é recebida ocorre uma multiplicação por uma matriz contendo pesos  $W$  e logo após é somada a tendência  $b$ , como mostrado a seguir

$$S = X * W + b \quad (2.5)$$

sendo  $S$  a saída.

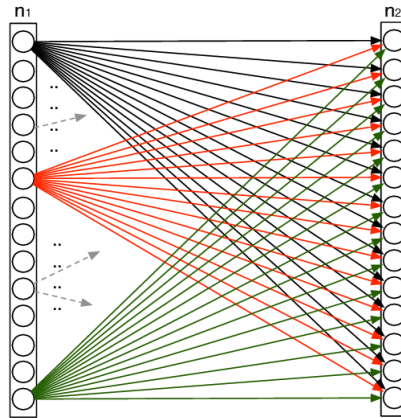


Figura 13 – Representação da camada totalmente conectada

Fonte: Wei Ma, Jun Lu, 2017

## 2.4 DETECÇÃO DE REGIÕES DE INTERESSE

As redes neurais profundas são ótimas para classificar imagens, porém elas não detectam qual região da imagem deve ser classificada. Para a detecção de regiões de interesse, algoritmos de visão computacional são utilizados.

### 2.4.1 Redes Neurais Convolucionais baseadas em regiões

As redes neurais convolucionais baseadas em regiões são conhecidas como R-CNN (Region-based Convolutional Neural Networks). A proposta desse algoritmo é dividir a imagem em sub-regiões, sendo elas retangulares ou quadradas. Após selecionadas, as dimensões das imagens são alteradas para que sejam aceitas como entrada de uma CNN. E como último passo, a rede neural classifica a região. (GIRSHICK et al., 2014)

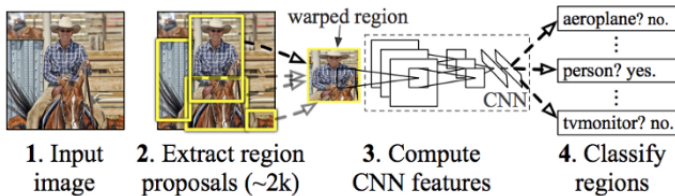


Figura 14 – Representação do algoritmo de RCNN

Fonte: GIRSHICK et al., 2014

Na figura 14, é representado os quatro passos de uma RCNN. Em 1 temos a imagem de entrada. Em 2 é mostrado a subdivisão da imagem de entrada em que normalmente são geradas em torno de duas mil sub imagens. A primeira ação do passo 3 é mudar as dimensões da sub imagem a ser analisada para as dimensões determinadas na criação da CNN, assim a imagem é analisada pela CNN e no passo 4 temos a classificação. (GIRSHICK et al., 2014)

### 2.4.2 Histogramas de gradientes orientados

O Histogramas de gradientes orientados, mais conhecido como HOG, é um modelo de detecção desenvolvido por Navneet Dalal and Bill Triggs, e inicialmente proposto para a detecção de pessoas.

O HOG é bastante usado pois a detecção de bordas da região de interesse é precisa. (MATLAB Documentation, 2017). Esse algoritmo é baseado na ideia de que o formato de um objeto pode ser caracterizado pela distribuição de intensidade de gradientes locais. A partir da intensidade desses gradientes em determinada direção é traçado um vetor. (DALAL; TRIGGS, 2005)

A extração dessa característica está exemplificada na figura 15.

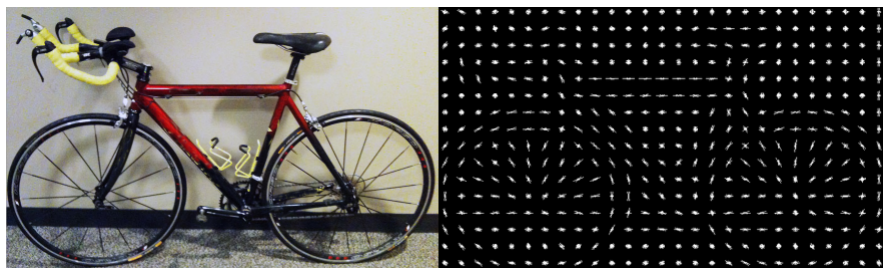


Figura 15 – Representação da saída do algoritmo de HOG  
Fonte: MATLAB Documentation, 2017



### 3 TRABALHOS CORRELATOS

Neste capítulo é abordado quatro trabalhos voltados para a área de detecção de placas de trânsito e o uso de Deep Learning.

#### 3.1 DETECÇÃO AUTOMÁTICA E RECONHECIMENTO DE SINALIZAÇÃO DE TRANSITO USANDO ANALISE DE ESTRUTURAS GEOMÉTRICAS

Esse trabalho é um artigo escrito por Vavilin Andrey e Kang Hyun Jo, da Universidade de Ulsan na Korea. O projeto visa reconhecer a sinalização de transito através de padrões geométricos em na imagem e após reconhecidos destacar as possíveis placas usando as cores padronizadas para cada tipo de sinalização. É usado filtros para fazer a separação das cores vermelha, verde e azul do sistema RGB, como mostrado na figura 16.

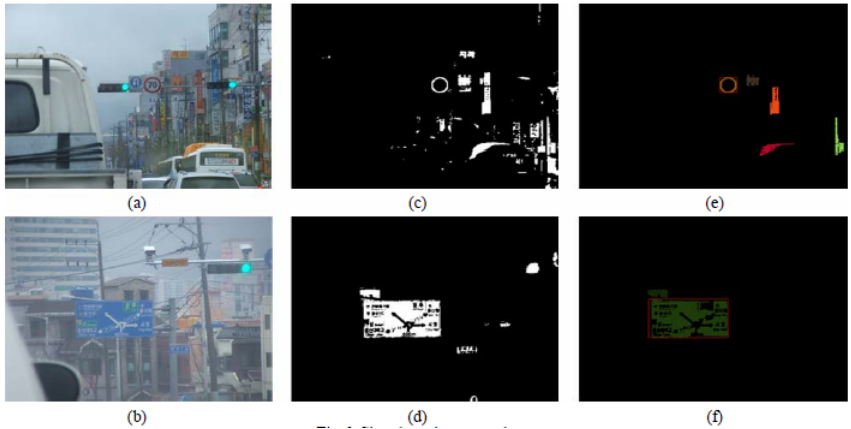


Figura 16 – Exemplo de detecção: (a,b) imagens de entrada; (c,d) detecção das possíveis regiões pelo formato; (e,f) possíveis áreas utilizando os filtros de cores.

Utiliza aprendizado de máquina para extrair essas características e realizar a análise dos padrões obtidos. Segundo a conclusão do autor, este método pode ser utilizado para a extração e detecção de placas em

tempo real, inclusive em cenários noturnos e com chuva.

### 3.2 DETECÇÃO E CLASSIFICAÇÃO DE SINALIZAÇÃO VERTICAL DE TRÂNSITO EM CENÁRIOS COMPLEXOS

Esse trabalho é uma dissertação de mestrado desenvolvida por Igor Gustavo Hoelscher, na Universidade Federal do Rio Grande do Sul (UFRGS).

Neste trabalho o autor realiza um estudo de técnicas de detecção e classificação de sinalização de trânsito em cenários de tráfego. São apresentadas duas abordagens para detecção, a primeira utiliza limiarização de cor em conjunto com descritores de Fourier, onde assume que um contorno fechado pode ser representado como uma função complexa contínua que torna possível obter os coeficientes de Fourier utilizando a transformada de Fourier

$$C(n) = \frac{1}{L} \int_{l=0}^L c(l) \exp\left(-\frac{i2\pi nl}{L}\right) dl \quad (3.1)$$

onde  $C$  é o contorno,  $L$  é o comprimento do contorno,  $T$  é o número de descritores,  $n = 0, \dots, N$ , onde  $N \leq T$ .

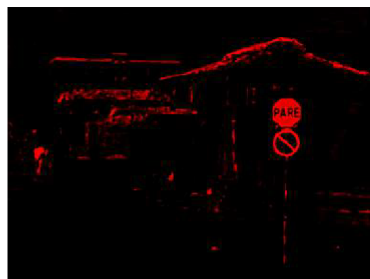


Figura 17 – Exemplo de Segmentação por Limiarização em cenários de trânsito da Alemanha.

Fonte: Igor Gustavo Hoelscher, 2017

A segunda abordagem utiliza um método de filtragem de cores baseado em Lógica Fuzzy. Essa lógica trabalha com valores entre completamente verdadeiro e completamente falso, é multivalorada podendo assumir uma verdade parcial. Para esta abordagem é realizado uma transformação da imagem RGB para um mapa em escala de cinza onde é destacado regiões de interesse pela cor. É criado um modelo de

fuzzificação para cada uma das cores de interesse.



(a) Segmentação da cor vermelha.



(b) Segmentação da cor amarela.

Figura 18 – Exemplos de segmentação por fuzzificação em cenários de trânsito do Brasil. Para cada cenário é apresentada sua imagem segmentada com valores de intensidade pintados da cor de interesse, para fins de ilustração.

Fonte: Igor Gustavo Hoelscher, 2017

Para a classificação, o autor criou redes neurais em Python 2.7, utilizando a biblioteca conhecida como TensorFlow. Foram usados como dados de entrada imagens de sinalização da Alemanha e do Brasil, e realizada a análise de diferentes redes e seus desempenhos.

O método para segmentação obteve 94% de acurácia, enquanto a classificação foi de 97%.

### 3.3 EXTRAÇÃO DE INFORMAÇÕES DE COR E FORMA PARA DETECÇÃO DE PLACAS DE TRÂNSITO EM IMAGENS

Esse trabalho é uma monografia escrita por Ricardo Cezar B. Rodrigues no Instituto Tecnológico de Aeronáutica (ITA). Nesse trabalho o autor realiza um pré-processamento, usando um filtro de suavização e logo em seguida aplica um filtro de cor amarela e um de vermelho afim de identificar regiões de interesse. Foi utilizado as componentes do modelo HSB , matiz e saturação, para destacar as possíveis áreas de interesse. A figura 19 mostra o resultado após esse pré-processamento.



Figura 19 – Detecção utilizando o modelo HSB

Foram selecionados quatro atributos que deviam ser levados em consideração, o fator da forma, a compacidade, a proporção e o eixo menor. Para realizar a classificação, foram testados dois modelos, redes bayesianas, no qual os atributos selecionados levam o algoritmo a escolher a categoria mais provável de saída, e KNN (K nearest neighbors, traduzido para K vizinhos mais próximos), onde a imagem é classificada por pesos em cada tipo de categoria.

No trabalho, apesar de usar algoritmos que também serviriam para classificação, o autor optou por não classificar pelo significado de cada placa, mas sim classificar as imagens de entrada em “Não-placa”, “Octógono”, “Losango”, “Triângulo” e “Circulo”.

Segundo o autor, os resultados do modelo HSB para detecção foram satisfatórios, obtendo cerca 90% de acertos.



### 3.4 REDES CONVOLUCIONAIS MUITO PROFUNDAS PARA RE-CONHECIMENTO DE IMAGENS EM LARGA ESCALA

Este trabalho correlato é um artigo publicado na ICLR de 2015, foi desenvolvido por Karen Simonyan e Andrew Zisserman, da Universidade de Oxford. Neste trabalho os autores fazem um estudo de seis diferentes arquiteturas de ConvNets, todas elas recebendo como entrada 1.3 milhão de imagens RGB de 224 pixels de altura por 224 de largura, e mil categorias de saída. Foi desenvolvido utilizando a ferramenta Caffe para C++.

A	A-LRN	B	C	D	E
Camada de entrada					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
MaxPool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
MaxPool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
MaxPool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
MaxPool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
MaxPool					
Completamente conectada					
Completamente conectada					
Completamente conectada					
SoftMax					

Tabela 1 – Camadas utilizadas para teste

Na tabela 1, é mostrada a arquitetura das redes desenvolvidas, no qual o primeiro parâmetro da camada convolucional é referente ao tamanho da matriz quadrada utilizada para a convolução, e o segundo é o número de filtros usados. A camada LRN realiza a normalização das entradas de sua camada por regiões. Cada entrada é dividida por

$$(1 + (\frac{\alpha}{n}) \sum_i x_i^2)^\beta \quad (3.2)$$

onde  $n$  é o tamanho da região e o somatório é dado pelo valor da região centralizada. (BERKELEY, 2018)

Apesar deste trabalho não ter utilizado placas de trânsito, foi importante para entender conceitos relacionados a camadas de uma rede neural.

## 4 METODOLOGIA PROPOSTA

A partir dos conceitos aprendidos na fundamentação teórica, as decisões de implementação foram separadas em 4 partes, sendo essas listadas a seguir:

### 4.1 DADOS DE ENTRADA

Este trabalho faz parte da categoria de aprendizado supervisionado, ou seja, tendo conjuntos de dados de entrada rotulados, se espera uma saída pertencente a um dos conjuntos (BARROS, 2016), as imagens foram separadas em 12 conjuntos rotulados, sendo esses listados a seguir:

- Velocidade Máxima permitida 30km/h
- Velocidade Máxima permitida 40km/h
- Velocidade Máxima permitida 60km/h
- Velocidade Máxima permitida 70km/h
- Dê a preferência
- Sentido Obrigatório Direita
- Sentido Obrigatório Esquerda
- Estacionamento regulamentado
- Parada Obrigatória
- Passagem Sinalizada de Pedestres
- Proibido estacionar
- Proibido Parar e Estacionar

Para evitar *overfitting*, foram usadas tanto placas em estado de conservação bom, quanto placas que estavam quebradas ou continham pichações.

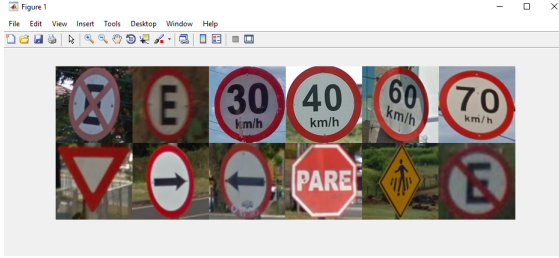


Figura 20 – Exemplo de imagens de entrada  
Fonte: Daniela Preto

## 4.2 CONFIGURAÇÕES E PARÂMETROS

As configurações de treino usadas nas CNNs estão citadas a seguir, e o valor de cada parâmetro, decididos através da realização de treinos, são mostrados no capítulo 5:

- SGDM - Stochastic Gradient Descent with Momentum

O gradiente estocástico descendente atualiza os pesos e os vieses a fim de minimizar a função de perda, dada a seguinte fórmula

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \quad (4.1)$$

onde  $l$  é o número de iterações,  $\alpha > 0$  é a taxa de aprendizado,  $E(\theta)$  é a função de perda e seu gradiente é  $\nabla E(\theta)$ .

Para este trabalho adicionaremos o *Momentum*, fazendo com que a oscilação da função de perda seja reduzida. Para isso, será adicionado  $\gamma$ , que será um auxílio do gradiente anterior, fazendo com que a fórmula fique da seguinte maneira (MATLAB Documentation, 2017).

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}). \quad (4.2)$$

- InitialLearnRate

A taxa de aprendizado é um parâmetro que controla o quanto o peso da rede pode ser ajustado. O uso de uma taxa muito baixa faz com que a função de perda tenha uma precisão muito boa, porém o tempo de treinamento aumentará consideravelmente. Entretanto, o uso de uma taxa muito alta, faz com que a rede tenha

um falso aprendizado, pois a função de perda convergirá a zero rapidamente, mas os valores dos pesos estarão aleatórios. (ZULKIFLI, 2018)

- LearnRateSchedule

Foi utilizado o algoritmo de 'piecewise', o qual faz com que a taxa de aprendizado seja atualizada a cada determinado período de treinamento (MATLAB Documentation, 2017).

- LearnRateDropFactor

É um número pré-determinado que multiplicará o fator de aprendizado no período escolhido (MATLAB Documentation, 2017).

- LearnRateDropPeriod

Determina o período em que a taxa de aprendizado será atualizada. (MATLAB Documentation, 2017).

- 'L2Regularization'

A regularização dos pesos da função de perda  $E(\theta)$  (*Loss Function*) é feita para reduzir o *overfitting*, que é um termo estatístico utilizado quando um modelo funciona muito bem para um conjunto de dados, porém é ineficaz para novos dados. Fazendo com que a função de perda fique da seguinte maneira

$$R(\theta) = E(\theta) + \gamma \frac{1}{2} w^T w \quad (4.3)$$

onde  $w$  é o peso e  $\gamma$  é o coeficiente de regularização

- 'Época'

*Epoch* é o período em que uma rede com *backpropagation* passa todos os dados de entrada do começo ao fim e do fim ao começo.

- Iteração

É o número de vezes em que um determinado número fixo de imagens de treinamento vão até o final da rede e voltam. Cada ida e volta, é considerada uma iteração.

## 4.3 ARQUITETURAS DE CNN

Foram testados diversos tipos de camadas com diferentes parâmetros. A tabela a seguir mostra seis tipos de redes que obtiveram resultados interessantes a serem comparados.

A	B	C	D	E	F
Camada de entrada					
conv 5-32	conv5-32	conv3-32	conv5-64	conv5-64	conv5-128
ReLU					
MaxPool					
conv 5-32	conv5-32	conv3-32	conv5-64	conv5-128	conv5-64
ReLU					
MaxPool					
conv 5-32	conv5-32	conv3-32	conv5-64		
ReLU	ReLU	ReLU	ReLU		
MaxPool	MaxPool	MaxPool	MaxPool		
conv 5-32					
ReLU					
MaxPool					
Completamente conectada					
ReLU					
Completamente conectada					
SoftMax					
Camada de classificação					

Tabela 2 – Redes neurais implementadas

A rede A possui 3 camadas a mais que a rede B, uma de convolução, uma ReLU e uma de *Pooling*. A adição dessas camadas fez com que a rede A demorasse em torno de uma hora e dez minutos para que o aprendizado chegasse a 99%, enquanto a rede B em 25 minutos chegou a um aprendizado convergindo em 100%. Observa-se que o fato de ter mais camadas não é obrigatoriamente um fator positivo.

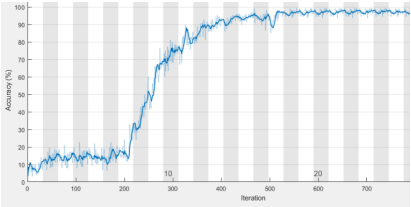


Figura 21 – Treinamento rede A

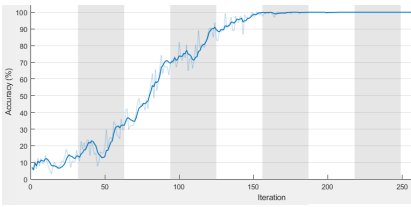


Figura 22 – Treinamento rede B

Comparando a rede B com a rede C, a sua diferença de implementação foi o tamanho da matriz de convolução. O aprendizado de

ambas convergiram em 100% porém a rede B, cuja matriz é maior, tem um aprendizado em torno de 8 minutos mais rápido.

Entre o treinamento da rede B com a rede D, foram aumentados os números de filtros das camadas convolucionais. O tempo de treinamento das duas foi bem parecido, a rede B convergiu em 100% com 4 minutos de antecedência, porém a rede D convergiu em 100% no *epoch* quatro, enquanto a B, apenas no *epoch* sete.

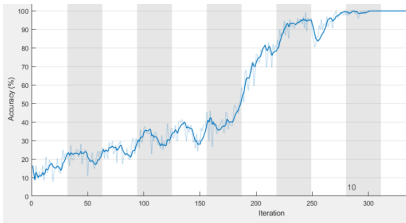


Figura 23 – Treinamento rede C

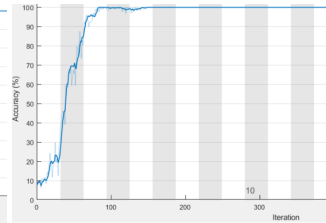


Figura 24 – Treinamento rede D

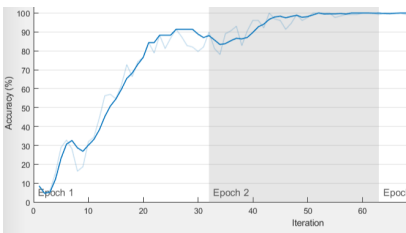


Figura 25 – Treinamento rede E

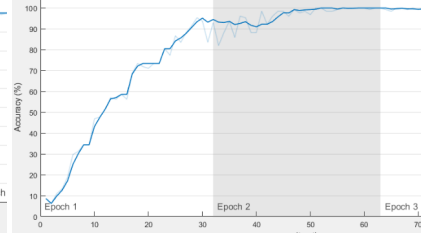


Figura 26 – Treinamento rede F

Tando para o treinamento da rede E, quanto o treinamento da rede F, o número de filtros foi aumentado significativamente. Com isso, foi observado que quanto maior o número de filtros menos *epoch* serão necessárias para ter um bom aprendizado, porém não necessariamente o tempo será menor.

Nas redes E e F, o aprendizado convergiu em 100% no *epoch* 3, como a rede E demorou apenas 22 minutos para ser treinada, enquanto a rede F demorou 45. A rede E foi escolhida para os próximos testes deste trabalho.

## 4.4 DETECÇÃO

Nesta seção é mostrado a pré configuração dos algoritmos de detecção propostos.

### 4.4.1 Treinamento da RCNN

Para o treinamento da RCNN foram utilizadas as mesmas imagens de entrada usadas para os treinamentos das CNNs. O treinamento foi feito a partir da função *'trainRCNNObjectDetector'* existente na biblioteca de visão computacional, enviando como parâmetros, as imagens, a CNN e as configurações a serem utilizadas. Na última camada completamente conectada de uma RCNN deve ser especificado como parâmetro o número de classes mais um, fazendo com que também seja possível que a área analisada não seja parte de nenhuma das classes.

### 4.4.2 Treinamento do HOG

Para o treinamento do algoritmo de HOG foram utilizadas outras imagens de entrada, imagens que possuísem não só a placa de trânsito. A partir dessas imagens foi utilizado o aplicativo 'Image Labeler' para ressaltar as áreas de interesse nas imagens, como mostrado na figura 27, onde as áreas de interesse estão marcadas em azul claro.



Figura 27 – Exemplos de imagens com regiões de interesse

Para que o algoritmo funcione, também são necessárias imagens chamadas de “negativo”, onde não possuem nenhuma área de interesse, tais como mostrado na figura 28.





Figura 28 – Exemplos de "negativos"

A figura 29 mostra exemplos de regiões de interesse após utilizar o HOG, são os mesmos tipos de placas mostrados na imagem 26. Na primeira é extraído um círculo, na segunda um losango e na terceira um octógono.

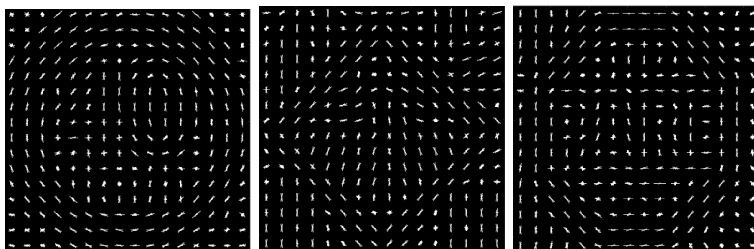


Figura 29 – Regiões de interesse após o algoritmo de HOG

Para realizar o treinamento foi utilizado a função “trainCascadeObjectDetector” que assim como a função de RCNN, faz parte da biblioteca de visão computacional. Essa função também pode ser utilizada para outros algoritmos de detecção, tais como Viola-Jones e Padrões Locais Binários. Como parâmetros são utilizadas as imagens que possuem áreas de interesse e os “negativos”.



## 5 IMPLEMENTAÇÃO

Nesse capítulo é explicado como foi feita a implementação, tal como os parâmetros usados.

### 5.1 CRIAÇÃO DE UMA CNN

A implementação das camadas de uma CNN em MatLab segue os conceitos apresentados na seção 2.3. Para tem uma maior clareza ao manipular cada camada, elas foram divididas em camadas de entrada, intermediárias e finais, representadas no algoritmo por ‘inputLayer’, ‘middleLayer’ e ‘finalLayers’ respectivamente, e depois combinadas em ‘layers’, como mostrado no algoritmo 1.

Os parâmetros da função ‘imageInputLayer’ são respectivamente a altura, a largura e número de canais, nesse trabalho por trabalhar com RGB, é usado o valor 3.

Em ‘convolution2dLayer’ há dois parâmetros obrigatórios, o primeiro representado é o tamanho do filtro que será utilizado e o segundo é o número de filtros. Nesse trabalho utilizamos o ‘Padding’, estabelecendo seu valor sendo 2.

A função ‘maxPooling2dLayer’ possui apenas um parâmetro obrigatório, sendo ele um número que determinará o valor da altura e largura na matriz quadrada utilizada. Foi utilizado também o ‘Stride’ e seu valor fixado como 2.

A camada ‘fullyConnectedLayer’ recebe como parâmetro o número de neurônios que terá de saída. Não existe número máximo de vezes para utilizar essa camada, porém o último uso deve ter como parâmetro o número de classes da rede, como mostrado na linha 13.

No algoritmo 2, temos a função ‘trainingOptions’, que recebe como parâmetros as configurações de treinamento na CNN. O conceito de cada item está explicado na seção 4.2. Em seguida, temos a função ‘trainNetwork’, onde ocorre o treinamento da rede, recebendo como parâmetros as imagens de entrada, as camadas pré-estabelecidas e as opções de configuração da rede.

Para que seja implementada uma RCNN, a função ‘trainNetwork’ deve ser trocada por ‘trainRCNNObjectDetector’, utilizando os mesmos parâmetros.

---

**Algorithm 1** Criação das camadas de uma CNN
 

---

```

inputLayer = imageInputLayer([128 128 3])
middleLayers = [
    convolution2dLayer(filterSize, 64, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(3, 'Stride', 2)
    convolution2dLayer(filterSize, 128, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(3, 'Stride', 2)
]
finalLayers = [
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(12)
    softmaxLayer
    classificationLayer

layers = [
    inputLayer
    middleLayers
    finalLayers
]

```

---



---

**Algorithm 2** Criação da CNN
 

---

```

opts = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'Plots', 'training-progress', ...
    'Verbose', true);
convnet = trainNetwork(Data, layers, opts);

```

---

## 5.2 DETECÇÃO DE IMAGENS E CLASSIFICAÇÃO

O algoritmo 3, foi utilizado para um teste inicial, não utilizando detecção, apenas testando a classificação. A imagem da câmera é capturada, e para que a função ‘classify’, onde é feita a classificação, funcione, a imagem é redimensionada com a função ‘mresize’

---

### Algorithm 3 Classificação de imagens capturadas

---

```

1: camera = webcam                                ▷ Captura a imagem da câmera
2: while true do
3:   imagem = camera.snapshot;
4:   imagem = imresize (imagem,[128,128]);
5:   label = classify (convnet,picture);            ▷ Classifica a imagem
   capturada
6:   image(imagem);                                ▷ Mostra a imagem capturada
7:   title(char(label));
8:   drawnow;
   end

```

---



---

### Algorithm 4 Detecção por RCNN e Classificação

---

```

1: while true do
2:   imagem = camera.snapshot;
3:   picture = imresize (picture,[altura,largura]);
4:   [bbox, score, label] = detect(rcnn, picture);
5:   annotation = sprintf('% s: (Confidence = %f)', label(idx),
   score);
6:   detectedImg = insertObjectAnnotation(picture, 'rectangle',
   bbox, annotation);
7:   image(detectedImg);
8:   drawnow;
   end

```

---

O algoritmo 4 é similar ao 3, porém a imagem capturada pela câmera pode ter quaisquer dimensões, pois ocorre a detecção da área de interesse antes da classificação.

Na linha 5, a função ‘detect’ tem como parâmetro a CNN treinada e a imagem a ser analisada, e seu retorno é uma matriz formada pela área da figura analisada, a pontuação de saída da classe e o nome da classe.

Na linha 6, é decidida a anotação que aparecerá na figura. Essa anotação é utilizada na linha 7, onde é inserida na imagem.

A detecção via HOG é mostrada no algoritmo 5. Na linha, é mostrado a função de treinamento recebendo as imagens com áreas de interesse e os negativos como já dito no capítulo anterior, porém o primeiro parâmetro é o nome do arquivo em que o treinamento será salvo.

Diferentemente do algoritmo de RCNN, quando esse treinamento é executado o retorno é apenas a área de interesse, como mostrado na linha 3.

Na linha 5, é salvo o número de regiões de interesse retornadas pelo treinamento. Da linha 6 à linha 10 é opcional, porém em um cenário com muitas regiões detectadas, é determinado um tamanho mínimo de interesse.

Da linha 12 à linha 16, o algoritmo percorre todas as áreas de interesse retornadas. Na linha 13 ocorre o recorte dessa região com a função ‘incrop’ que recebe como parâmetros a imagem inicial e as coordenadas que deve ser feito o recorte. Na linha 14, ocorre o redimensionamento da imagem recortada para que na linha 15 seja feita a classificação com a CNN desejada.

Como saída, temos a imagem de entrada com marcação em todas as regiões de interesse, como mostrado na linha 17.

---

**Algorithm 5** Detecção por HOG e Classificação
 

---

```

1: trainCascadeObjectDetector('placasDetector.xml',positivos, nega-
   positivos);
2: while true do
3:   bboxes = step(placasDetector,imagem);
4:   i=1
5:   sizebbboxes = size(bboxes,1)
6:   dim = bboxes(i,3) * bboxes(i,4)
7:   if (dim < dimMinimo)
8:     bboxes(i,:) = []
9:   EndIf
10:  sizebbboxes = size(bboxes,1)
   end
11: j=1
12: while j : sizebbboxes do
13:   c = imcrop(imagem,bboxes(j,:))
14:   c = imresize(c,[128,128])
15:   class = classify ( convnet ,c )
16:   j=j+1
17: output = insertObjectAnnotation(imagem,'rectangle',bboxes,class);

```

---





## 6 RESULTADOS

Alguns resultados já foram mostrados na seção 4.3, pois a partir desses foi escolhida a CNN usada nos demais passos. Nesse capítulo mostraremos os resultados finais da implementação.

### 6.1 RESULTADOS DE CLASSIFICAÇÃO

Para ter uma melhor visualização dos resultados de classificação da CNN, a imagem abaixo mostra a matriz de confusão. Na horizontal temos a classe esperada para a classificação, e na vertical temos a classificação dada pela rede neural.

		Confusion Matrix												
Output Class		30	40	60	70	DeAPreferencia	Direita	Esquerda	EstacionamentoRegulamentado	Pare	PassagemSinalizadaDePedestres	ProibidoEstacionar	ProibidoPararEEstacionar	
	30	68 8.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	40	0 0.0%	68 8.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	60	0 0.0%	0 0.0%	68 8.4%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.6% 1.4%
	70	0 0.0%	0 0.0%	0 0.0%	63 7.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	DeAPreferencia	0 0.0%	0 0.0%	0 0.0%	0 0.0%	61 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	Direita	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	64 7.9%	4 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.1% 5.9%
	Esquerda	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.5%	63 7.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.0% 6.0%
	EstacionamentoRegulamentado	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	68 8.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
	Pare	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	68 8.4%	0 0.0%	1 0.1%	0 0.0%	98.6% 1.4%
	PassagemSinalizadaDePedestres	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	69 8.5%	1 0.1%	0 0.0%	98.6% 1.4%
	ProibidoEstacionar	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	66 8.1%	0 0.0%	100% 0.0%
	ProibidoPararEEstacionar	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	72 8.9%	100% 0.0%
		100% 0.0%	100% 0.0%	100% 0.0%	97.1% 2.9%	100% 0.0%	94.1% 5.9%	94.0% 6.0%	100% 0.0%	100% 0.0%	97.1% 2.9%	100% 0.0%	98.7% 1.3%	
		30	40	60	70	DeAPreferencia	Direita	Esquerda	EstacionamentoRegulamentado	Pare	PassagemSinalizadaDePedestres	ProibidoEstacionar	ProibidoPararEEstacionar	
		Target Class												

Figura 30 – Matriz de confusão

De todas as imagens de teste, 98.7% foram classificadas corretamente, e 1% dos erros ocorreram nas classes "Direita" e "Esquerda", o que pode ter sido causado pelo fato de ambas serem placas de "Sentido Obrigatório" porém em diferentes direções.

Também foram realizados testes de classificação com a câmera do computador, no qual o resultado foi muito satisfatório, porém as placas de teste deveriam estar como alvo principal da câmera, cobrindo o máximo dos 128 x 128 pixels da entrada da CNN, como mostrado na figura 31, o que nos leva a implementações de detecção de áreas de interesse.

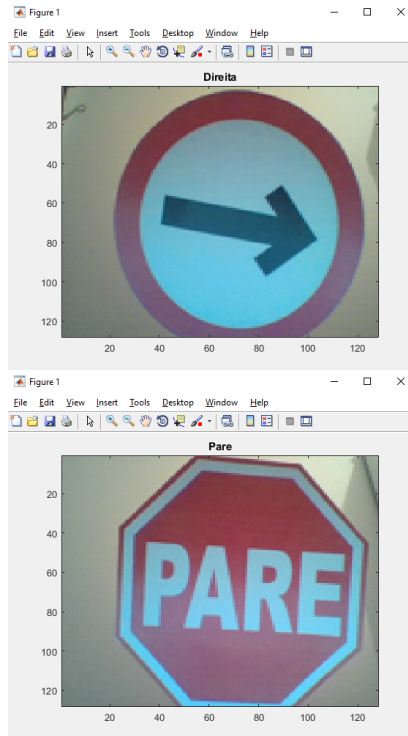


Figura 31 – Exemplos de saída do algoritmo 3.

## 6.2 RESULTADOS DE CLASSIFICAÇÃO COM DETECÇÃO

O primeiro algoritmo a ser testado foi o de RCNN. Obteve um resultado satisfatório quando testado em fotos como entrada, como mostrado na figura 32, porém é um algoritmo demorado para analisar cada frame, podendo levar alguns segundos, o que acabou por ter um desempenho ruim para imagens de tempo real e vídeos.



Figura 32 – Teste de RCNN com foto

Na figura 33 é mostrado a captura de um frame, utilizando a câmera em tempo real, e mesmo em um cenário simples o retângulo a ser desenhado não carrega perfeitamente.

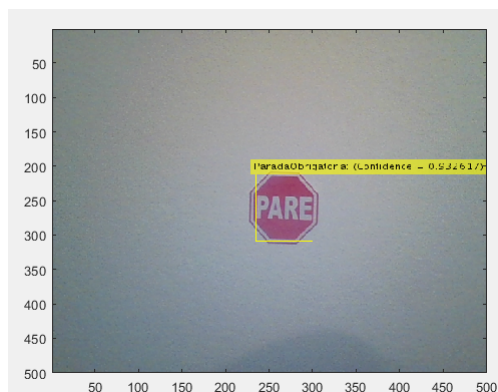


Figura 33 – Teste de RCNN com imagem em tempo real

O segundo algoritmo testado foi o HOG, e seu resultado foi indiscutivelmente melhor do que a detecção com RCNN, porém não perfeito a ponto de ser usado em um sistema de tempo real.

Na figura 34, temos a captura de tela onde a entrada do algoritmo é um vídeo. O algoritmo analisou corretamente as duas regiões de interesse. A placa de “Parada Obrigatória” foi classificada perfeitamente pela CNN, porém como não existe uma classe de saída com o significado “Proibido Virar a Esquerda”, a região de interesse foi classificada erroneamente como “Proibido Estacionar”.

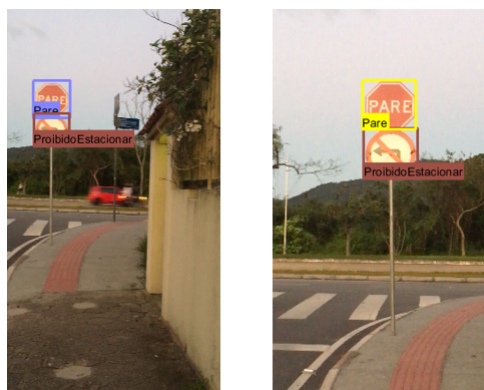


Figura 34 – Teste de detecção usando HOG

## 7 CONCLUSÃO

O presente trabalho buscou analisar o funcionamento de redes neurais em uma das suas mais diversas funções, a classificação de imagens.

Na leitura inicial sobre trabalhos que visassem a classificação de imagens, ficou clara a necessidade de algoritmos de visão computacional para a detecção de áreas de interesse. Revisando a literatura foi mostrada a importância de cada camada para a implementação de uma CNN.

Durante a realização dos testes de criação das redes neurais artificiais, ficou claro que mesmo o MatLab sendo um software excelente para se trabalhar com imagens, é de extrema importância o poder de processamento que o hardware deve ter para trabalhar com Deep Learning.

As CNNs profundas tiveram um ótimo desempenho para a classificação, porém para datasets pequenos, outras técnicas de Machine Learning podem obter um resultado parecido, porém com um tempo menor e utilizando menos processamento. Em comparação com os trabalhos correlatos, o trabalho mostrado na sessão 3.2 obteve o melhor resultado, tendo uma acurácia alta tanto para detecção quanto para classificação, podendo ter influência do uso da biblioteca TensorFlow e da linguagem Python, sendo essas mais "leves" que o software MatLab. Apesar do sucesso em detectar e classificar imagens corretamente, ainda se deve focar na melhoria dos algoritmos para que consigam ser usados em tempo real.

### 7.1 TRABALHOS FUTUROS

A tendência da área de aprendizado de máquina é continuar evoluindo suas técnicas, porém focando na subárea deste trabalho, deve-se futuramente aumentar o banco de imagens de entrada, para que abranja as dezenas de outros tipos de placas de sinalização. Outro fator importante é propor outros algoritmos de extração de características para realizar a detecção de maneira mais rápida.

Apesar de ter obtido resultados satisfatórios, realizar mais testes de implementação de CNNs, com diferentes tipos de camadas é essencial.



## REFERÊNCIAS

ANDREY, K. Machine learning applications in cell image analysis. *Immunology and Cell Biology*, v. 95, n. 6, p. 525–530, 2017.  
<<https://onlinelibrary.wiley.com/doi/abs/10.1038/icb.2017.16>>.

BARROS, P. *Aprendizagem de Maquina: Supervisionada ou Não Supervisionada?* 2016.  
<<https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-nao-supervisionada-7d01f78cd80a>>. Acessado em 01/08/2018.

BERKELEY, A. I. R. *Local Response Normalization (LRN)/ Caffe Documentation*. 2018.  
<<http://caffe.berkeleyvision.org/tutorial/layers/lrn.html>>.

CHEN, X.-W.; LIN, X. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2014.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. Rhone-Alps, 655 avenue de l'Europe, Montbonnot 38334, France, 2005.

DENATRAN. *Anuários estatísticos do DENATRAN*. 2000.

DETRAN. *Convenção Internacional sobre Trânsito Viário*. 1978.  
<<http://www.detran.pr.gov.br/arquivos/File/legislacao/outraslegislacoesestaduan.pdf>>.

FAUSETT, L. *Fundamentals Of Neural Networks, 1st ed.* [S.l.]: Pearson, 1994.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. United States, 2014.

HAYKIN, S. *Neural networks and learning machines, 3rd ed.* Ontario, Canada: Pearson, 2009.

JANAI, J. et al. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. 2017.  
<<http://adsabs.harvard.edu/abs/2017arXiv170405519J>>.

MATLAB, I.; SIMULINK. *Introducing Deep Learning with MATLAB*. [S.l.], 2017.

NIELSEN, M. A. *Neural Networks And Deep Learning*. 2015. <<http://neuralnetworksanddeeplearning.com>>. Acessado em 01/06/2018.

ORGANIZATION, W. H. *Global Status Report On Road Safety*. 2015. <[https://www.who.int/violence\\_injury\\_prevention/road\\_safety\\_status/2015/en/](https://www.who.int/violence_injury_prevention/road_safety_status/2015/en/)>.

PACHECO, A. Redes neurais convolutivas. Brasil, 2017.

PEREIRA, T. *Qual a diferença entre Machine Learning e Deep Learning?* 2017. <<https://computerworld.com.br/brandpost/qual-diferenca-entre-machine-learning-e-deep-learning/>>. Acessado em 01/06/2018.

RUSSELL, S.; NORVIG, P. *Inteligencia Artificial, 3a edição*. Rio de Janeiro, Brazil: Elsevier, 2013.

ZAMBIASI, S. P. *Arquitetura das Redes Neurais*. Brasil: [s.n.], 2008. <<https://www.gsigma.ufsc.br/popov/aulas/rna/arquitetura/>>.

ZANNI, M. *Carros Inteligentes*. 2014. <<https://revistagalileu.globo.com/Revista/noticia/2014/01/carros-inteligentes.html>>. Acessado em 01/06/2018.

ZULKIFLI, H. *Understanding Learning Rates and How It Improves Performance in Deep Learning*. 2018. <<https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10>>. Acessado em 01/08/2018.



## APÊNDICE A – Fontes



---

```

Datasetpath = fullfile('C:', 'Users', 'Desktop', 'Image')
    = imageDatastore(Datasetpath, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');
inputLayer = imageInputLayer([128 128 3])
filterSize = [5 5];
numFilters = 32;
middleLayers = [
    convolution2dLayer(filterSize, numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(3, 'Stride', 2)
    convolution2dLayer(filterSize, numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(3, 'Stride', 2)
    convolution2dLayer(filterSize, 2 * numFilters, 'Padding', 2)
    reluLayer()
    maxPooling2dLayer(3, 'Stride', 2)
]
finalLayers = [
    fullyConnectedLayer(64)
    reluLayer
    fullyConnectedLayer(12)
    softmaxLayer
    classificationLayer
]
layers = [
    inputLayer
    middleLayers
    finalLayers
]
opts = trainingOptions('sgdm', ...
    'Momentum', 0.9, ...
    'InitialLearnRate', 0.001, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', 0.1, ...
    'LearnRateDropPeriod', 8, ...
    'L2Regularization', 0.004, ...
    'MaxEpochs', 9, ...
    'MiniBatchSize', 128, ...
    'Plots', 'training-progress', ...
    'Verbose', true);
convnet = trainNetwork(Data, layers, opts);
c = trainRCNNObjectDetector(d, l, opts, 'NegativeOverlapRange',
    [0 0.3]);

```

---

---

```

filenames = gTruth.DataSource.Source
rois = gTruth.LabelData
placas = horzcat(filenames, rois)
positivos = placas(:,1:2);
negativeFolder = fullfile('C:');
negativeImages = imageDatastore(negativeFolder);
trainCascadeObjectDetector('placasDetector.xml',positivos, ...
negativeFolder,'FalseAlarmRate',0.1,'NumCascadeStages',20);
placasDetector = vision.CascadeObjectDetector('placasDetector.xml');
bboxes = step(placasDetector,a);
i=1
sizebbboxes = size(bboxes,1)
while i < sizebbboxes
    altura= bboxes(i,3)
    largura= bboxes(i,4)
    tam = altura * largura
    if (tam < 9000)
        bboxes(i,:) = []
    end
    if(sizebbboxes == size(bboxes,1))
        i = i+1
    end
    sizebbboxes = size(bboxes,1)
end
j=1
while j:sizebbboxes
    c = imcrop(a,bboxes(j,:))
    c = imresize(c,[227,227])
    output = classify ( conv ,c )
    j=j+1
I = insertObjectAnnotation(a,'rectangle',bboxes,label);
figure, imshow(I);

```

---

## APÊNDICE B – Artigo



# Reconhecimento de Placas de Trânsito Por Meio de Deep Learning

Daniela O. Preto

Instituto de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) Caixa Postal 476 – 88.040-900–Florianópolis – SC – Brasil

daniela.preto@ufsc.br

**Abstract.** *This project will focus on the subgroup of artificial intelligence known as Deep Learning. This subgroup is a working method developed from the advance of neuroscience, where the learning system of the machine is based on what is known of the nervous system, trying to compute input data as if it were stimulus of a brain, and analyzing the responses that will be generated in each case.*

*In this paper was used this method for the recognition of traffic signs, that the pixels of the image are analyzed and then classify it. In the many layers will be considered features such as shape and colors. The final objective is to be able to analyze and classify correctly the images, even if they have a difference in brightness or a different background.*

**Resumo.** *Este projeto terá como foco um subgrupo da inteligência artificial conhecido como Deep Learning (Aprendizagem Profunda). Este subgrupo é um método de trabalho desenvolvido a partir do avanço da neurociência, onde o sistema de aprendizado da máquina é baseado no que se conhece do sistema nervoso humano, tentando computar dados de entradas como se fossem estímulos de um cérebro, e analisando as respostas que serão geradas em cada caso.*

*Neste artigo este método foi usado para o reconhecimento de placas de trânsito, fazendo com que fossem analisados os pixels da imagem para assim classificá-la. Nas diversas camadas serão analisadas características como o formato e as cores. O objetivo final é poder analisar e classificar corretamente as imagens, mesmo que tenham diferença de luminosidade ou um fundo diferente.*

**1. Introdução** Com o aumento significativo de automóveis ao redor no mundo, a sinalização de trânsito ganhou um papel importante para evitar incidentes automobilísticos (ORGANIZATION, 2015). As placas de trânsito foram criadas para que tenham fácil visibilidade, legibilidade

e precisão, de maneira que o motorista consiga facilmente saber o que deve ser feito (DENATRAN, 2000).

Segundo o observatório Nacional de Segurança Viária, 90 % dos acidentes são causadas por falhas humanas, fazendo com que cada vez mais se invistam em sistemas inteligentes para automóveis (ZANNI, 2014).

Desde a segunda metade do século passado ouvimos falar de inteligência artificial. (RUSSELL; NORVIG, 2013) Um tema que era tratado como futurista em filmes e documentários se tornou realidade, sendo hoje um assunto bastante comentado no mundo todo. Três fatores foram decisivos para estarmos no patamar atual, o avanço do poder de processamento dos computadores, a grande quantidade de dados disponíveis e o desenvolvimento de técnicas e algoritmos. (CHEN; LIN, 2014)

Este trabalho fala especificadamente do terceiro fator e como a partir destas técnicas e algoritmos podemos realizar o reconhecimento de imagens, especificamente de placas de trânsito, através de Deep Learning, podendo assim auxiliar pessoas quando estiverem dirigindo.

## 2. Fundamentação Teórica

A fundamentação teórica deste artigo foi dividida em duas partes, classificação e detecção de imagens.

### 2.1. Classificação

**Redes Neurais Artificiais:** Uma rede neural é uma máquina projetada para modelar a maneira pela qual o cérebro executa uma tarefa ou função particular de interesse. (HAYKIN, 2009) Ela pode possuir uma ou mais camadas, e cada neurônio de cada camada é responsável pelo aprendizado final.(ZAMBIASI, 2008)

**Deep Learning:** Dos algoritmos de aprendizado de máquina existentes, o que mais tem se destacado são as redes neurais artificiais profundas, ou também conhecidas como *Deep Learning*, pois não necessitam de extração manual de características. É um modelo em que o aprendizado é de início ao fim, a partir de dados de entrada brutos.(MATLAB Documentation, 2017).

**Redes Neurais Convolucionais:** Como uma RNA tradicional as redes neurais convolucionais também possuem neurônios com pesos, vieses e funções não-lineares, porém possui grande vantagem quando utilizada em imagens, por possuir o poder de codificar propriedades, faz com que o treinamento diminua e a precisão seja mantida. (PACHECO,



2017)

## 2.2. Detecção

**Redes Neurais Convolucionais baseadas em regiões:** A proposta desse algoritmo é dividir a imagem em sub-regiões, sendo elas retangulares ou quadradas. Após selecionadas, as dimensões das imagens são alteradas para que sejam aceitas como entrada de uma CNN. E como último passo, a rede neural classifica a região. (GIRSHICK et al., 2014)

**Histogramas de gradientes orientados:** O HOG é bastante usado pois a detecção de bordas da região de interesse é precisa. (MATLAB Documentation, 2017). Esse algoritmo é baseado na ideia de que o formato de um objeto pode ser caracterizado pela distribuição de intensidade de gradientes locais. A partir da intensidade desses gradientes em determinada direção é traçado um vetor. (DALAL; TRIGGS, 2005)

## 3. Modelo Proposto

Como a fundamentação teórica, o modelo proposto também é dividido em classificação e detecção de imagens.

### 3.1. Classificação

Para o desenvolvimento do artigo, foram testadas inúmeras CNNs, porém para a realização dos testes finais, a rede descrita abaixo foi escolhida devido a sua boa performance em relação às outras.

---

#### Algorithm 1 Criação das camadas de uma CNN

---

```
imageInputLayer([128 128 3])
convolution2dLayer(filterSize, 64, 'Padding', 2)
reluLayer()
maxPooling2dLayer(3, 'Stride', 2)
convolution2dLayer(filterSize, 128, 'Padding', 2)
reluLayer()
maxPooling2dLayer(3, 'Stride', 2)
fullyConnectedLayer(64)
reluLayer
fullyConnectedLayer(12)
softmaxLayer
classificationLayer
```

---

### 3.2. Detecção

Para o treinamento da RCNN foram utilizadas as mesmas imagens de entrada usadas para os treinamentos das CNNs. O treinamento foi feito a partir da função 'trainRCNNObjectDetector' existente na biblioteca de visão computacional, enviando como parâmetros, as imagens, a CNN e as configurações a serem utilizadas.

Para o treinamento do algoritmo de HOG foram utilizadas outras imagens de entrada, imagens que possuísem não só a placa de trânsito, mas também outras informações de fundo. A partir dessas imagens foi utilizado o aplicativo 'Image Labeler' para ressaltar as áreas de interesse nas imagens. Para que o algoritmo funcione, também são necessárias imagens chamadas de "negativo", onde não possuem nenhuma área de interesse.

## 4. Resultados

Para ter uma melhor visualização dos resultados de classificação da CNN, a imagem abaixo mostra a matriz de confusão. Na horizontal temos a classe esperada para a classificação, e na vertical temos a classificação dada pela rede neural. De todas as imagens de teste, 98.7% foram classificadas corretamente, e 1% dos erros ocorreram nas classes "Direita" e "Esquerda", o que pode ter sido causado pelo fato de ambas serem placas de "Sentido Obrigatório" porém em diferentes direções.

		Confusion Matrix												
Output Class		30	40	60	70	DeAPreferencia	Direita	Esquerda	EstacionamentoRegulamentado	Pare	PassagemSinalizadaDePedestres	ProibidoEstacionar	ProibidoPararEstacionar	
		68 8.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%	
		0 0.0%	68 8.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%	
		0 0.0%	0 0.0%	68 8.4%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	88.6% 1.4%	
		0 0.0%	0 0.0%	0 0.0%	63 7.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	61 7.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	64 7.9%	4 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.1% 5.9%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.5%	63 7.7%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.0% 6.0%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	68 8.4%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	68 8.4%	1 0.1%	0 0.0%	88.6% 1.4%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	69 8.5%	1 0.1%	88.6% 1.4%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	66 8.1%	100% 0.0%	
		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	72 8.9%	
		100% 0.0%	100% 0.0%	100% 0.0%	97.1% 2.9%	100% 0.0%	94.1% 5.9%	94.0% 6.0%	100% 0.0%	100% 0.0%	97.1% 2.9%	100% 0.0%	98.7% 1.3%	
		30	40	60	70	DeAPreferencia	Direita	Esquerda	EstacionamentoRegulamentado	Pare	PassagemSinalizadaDePedestres	ProibidoEstacionar	ProibidoPararEstacionar	
		Target Class												

O algoritmo de R-CNN obteve um resultado satisfatório quando testado em fotos como entrada, como mostrado na figura abaixo, porém é um algoritmo demorado para analisar cada *frame*, podendo levar alguns segundos, o que acabou por ter um desempenho ruim para imagens de tempo real e vídeos.



O segundo algoritmo testado foi o HOG, e seu resultado foi indiscutivelmente melhor do que a detecção com RCNN, porém não perfeito a ponto de ser usado em um sistema de tempo real. Na figura abaixo, temos a captura de tela onde a entrada do algoritmo é um vídeo.

O algoritmo analisou corretamente as duas regiões de interesse. A placa de “Parada Obrigatória” foi classificada perfeitamente pela CNN, porém como não existe uma classe de saída com o significado “Proibido Virar a Esquerda”, a região de interesse foi classificada erroneamente como “Proibido Estacionar”.



## 5. Conclusões

O presente trabalho buscou analisar o funcionamento de redes neurais em uma das suas mais diversas funções, a classificação de imagens.

Durante a realização dos testes de criação das redes neurais artificiais, ficou claro que mesmo o MatLab sendo um software excelente para se trabalhar com imagens, é de extrema importância o poder de processamento que o hardware deve ter para trabalhar com Deep Learning.

As CNNs profundas tiveram um ótimo desempenho para a classificação, porém para datasets pequenos, outras técnicas de *Machine Learning* podem obter um resultado parecido, porém com um tempo menor e utilizando menos processamento. Apesar do sucesso em detectar e classificar imagens corretamente, ainda se deve focar na melhoria dos algoritmos para que consigam ser usados em tempo real.

## Referências

- DENATRAN. Anuários estatísticos do DENATRAN. 2014.
- ORGANIZATION, W. H. Global Status Report On Road Safety. 2015. <<https://www.who.int/violenceinjuryprevention/roadsafetystatus/2015/en/>>
- ZANNI, M. Carros Inteligentes. 2014. <<https://revistagalileu.globo.com/Revista/noticia/2014/01/carros-inteligentes.html>>. Acessado em 01/06/2018
- PACHECO, A. Redes neurais convolutivas. Brasil, 2017.
- ZAMBIASI, S. P. Arquitetura das Redes Neurais. Brasil: [s.n.], 2008. <<https://www.gsigma.ufsc.br/popov/aulas/rna/arquitetura/>>.
- MATLAB, I.; SIMULINK. Introducing Deep Learning with MATLAB. [S.l.], 2017.
- DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. Rhone-Alps, 655 avenue de l'Europe, Montbonnot 38334, France, 2005
- CHEN, X.-W.; LIN, X. Big data deep learning: Challenges and perspectives. IEEE Access, 2014
- RUSSELL, S.; NORVIG, P. Inteligencia Artificial, 3a edição. Rio de Janeiro, Brazil: Elsevier, 2013